# TheFundsChain – Certificate Authorities

Certificate Authorities – a tentative setup. DLT Technology: Hyperledger fabric.

Reference: <http://hyperledger-fabric-ca.readthedocs.io/en/latest/users-guide.html>

## Role of a Certificate Authority

A CA delivers certificates in order to:

* Registering identity credentials (X509 certificate)
* Issue E-Certs for MSP
* Issue T-Certs
* Revoke / Renew these certificate

Cas may define a hierarchy of parent CA and subordinate CA’s.

Root CA: administered by caretaker

* Defines jurisdiction domains (“affiliations”)
* Registers and enrolls regulators as identities with a CA registrar role
* Populated attribute: TheFundsChain.Roles
* Identity types managed:

Distributor CA (for any peer with distribution role)

* Identity types managed: investors

Identity types:

* user: an authenticated user used to deliver certificates
* peer: E-Cert for chaincode peers
* client: E-Cert for SDK-only nodes
* admin: administrate an organization
* auditor: E-Cert for query only nodes
* caretaker: administrates the root CA domain
* validator: E-Cert for ordering-only peers
* investor: E-Cert for non-member identities (investors)
* investorOwner: administrate distribution domain

Use case:

* caretaker deploys the root CA: TheFundsChain Root CA Authority
* caretaker registers then enroll amf as jurisdictionOwner, then delegates to amf the French domain
  + amf deploys an intermediate CA : « TheFundsChain French CA Authority”
  + amf enrolls CACEIS France, BPSS France, SGSS France, etc… as organizations
  + amf enrolls NAM, AMUNDI, AXAIM, BNPIP, etc… as organizations
  + organizations deploy their own intermediate CA affiliated to the French CA
    - e.g. « TheFundsChain CACEIS France CA Authority”
    - CACEIS enrolls its nodes as peers
    - NAM enrolls its nodes as peers
  + distributors: amf enrolls NAM-DIRECT, NGAM as organizations
  + NAM-DIRECT, NGAM, … (distributors) deploy their own intermediate CA affiliated to the French CA
    - NAM-DIRECT enrolls its investors



# Certificate delivery

Sequence:

Registering regulators:

1. amf requests a CA certificate for identity type ‘jurisdictionOwner’ from root CA [CSR with CA profile] (path length: 1). Caretaker fullfils the request by registering the amf identity with adequate Registrar.Roles attribute

fabric-ca-client register \

--id.name amf --id.type jurisdictionOwner \

--id.affiliation thefundschain.fr \

--id.attrs 'hf.Revoker=true,\

hf.Registrar.Roles: "user,client,peer,validator,investor,auditor,organizationOwner, investorOwner",'

1. amf register its nodes as MSP with identity type ‘peer’
   1. TheFundsChain.Roles attribute is set to ‘regulator’

Registering organizations

1. Caceis France requests from amf an identity of type ‘organizationOWner’
2. Amf fullfils the request by registering Caceis under affiliation fr : thefundschain.fr.caceis
3. Caceis certificate is set with TheFundsChain attribute: ‘custodian’, ‘accountant’, …

(corresponds to functional roles validated by the regulator)

1. Now Caceis must register an identity its own nodes as MSP

**ISSUE HERE: CACEIS SHOULD NOT BE ABLE TO SET ATTRIBUTES AT WILL**

1. Caceis registers its own nodes on his CA with identity type ‘peer’
   1. TheFundsChain.Roles attribute set to ‘custodian’, etc..

# Manipulating E-Certs in Fabric

# Understanding attributes propagation with certificates authorities

Since attributes represent privileges, it is essential that an intermediate CA cannot provide certificate with any kind of attribute.

The idea is that an intermediate CA should be able to set an attribute only if already has this attribute set.

Example:

* only caretaker user on root CA may deliver a regulator attribute
* only amf user on French CA may delivery an organization a ‘custodian’ attribute
* the organization may set for its own peers the attribute set that their own CA has been given by amf

**Unfortunately, this is not the case with Fabric CA: every CA may deliver the attributes of its own choosing…**

**Note: these checks are specific to Fabric CA**

**They are implemented here:** <https://github.com/hyperledger/fabric-ca/blob/master/lib/serverregister.go>

**And could be augmented.**

**Exploring CA's out of the Hyperledger small world**

* **alleged best open source CA available off-the-shelf: EJBCA (**<https://www.ejbca.org/>**)**
* **almost complete PKI**
* **Signature Server: SignServer:** <https://www.signserver.org/>
* **Support: Certificate Revocation List an OCSP**
* **Java + HSQLDB (or other SQL back-end : e.g MySQL)**
  + **HSM support**
* **Almost complete because it does not support attribute certs**
* **For such certs, we need some SPKI software…**
* **The problem is that SPKI (i.e. delivering Attribute Certificates) is not implemented by major CA software**
  + **Separate tools exist. Many are poorly maintained/inactive projects**
  + **Regarding CA, Fabric CA may do at least for testing, if reasonably workable (always good to have a docker-ready brick at hand)**
  + **Strongswan pki (for linux):** pki --acert
  + **https://wiki.strongswan.org/projects/strongswan/wiki/IpsecPkiAcert**
  + **Okay for experimenting: at least it builds certs. Could even be called from a chaincode (e.g. golang call unix command, before we may port it natively as golang (for instance by porting the strongswan api in golang…)**
    - **Could be part of the "Party onboarding workflow" chaincode (PARTY)**
  + **How about VERIFYING attributes from within a golang module?**
    - **Golang package spki: a low level API**
    - <https://github.com/eadmund/spki/blob/master/design.md>
      * **Create stuff could be used to create "contractual authorizations"**
    - **Example acert validator (strongswan acert plugin):**

<https://wiki.strongswan.org/projects/strongswan/repository/revisions/master/entry/src/libstrongswan/plugins/acert/acert_validator.c>

**OKAY**

**Idée Générale:**

* **Fabriquer des e-Certs (à partir de Fabric CA par ex. ou sinon voir ce que cela donne avec EJBCA)**
* **Fabriquer localement des a-Certs à partir de la commande pki de strongswan**
* **Implémenter un validateur de a-Cert qui puisse vérifier la présence d'un attribut**

**Où puis je stocker mes a-certs?**

**Les S-expressions dans les a-certs c'est quoi?**

<https://en.wikipedia.org/wiki/Simple_public-key_infrastructure>